UDACITY

‹ Return to Classroom

Create Your Own Image Classifier - TensorFlow

审阅

代码审阅

HISTORY

Meets Specifications

Great work on project implementation!!! With this project you have gained in-depth knowledge on image classification using tensorflow framework. This project is starting point in the world of deep learning. Very impressive implementation and it shows amount of hard work and time you have committed to complete it.

Good work on notebook and code implementation, one improvement area I would recommend in code files is to do proper commenting so that code becomes more readable. You can further improve accuracy and reduce variance through hyperparameter tuning and data augmentation.

Great job!!! If this is your final project then congratulations for finishing this nanodegree program. All the best for future endevours!!!

Files Submitted

The submission includes all required files. (Model checkpoints not required.)

Well done including all required files for submission.

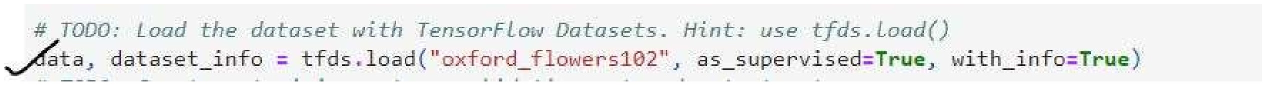
Part 1 - Development Notebook

All the necessary packages and modules are imported at the beginning of the notebook.

Well done including all necessary import at top section of the notebook.

The Oxford Flowers 102 dataset is loaded using TensorFlow Datasets.

Good work downloading oxford flowers!!



```
# TODO: Load the dataset with TensorFlow Datasets. Hint: use tfds.load()
data, dataset_info = tfds.load("oxford_flowers102", as_supervised=True, with_info=True)
```

The dataset is divided into a training set, a validation set, and a test set.

Well done retrieving train, validation and test data from dataset.

The number of examples in each set and the number classes in the dataset are extracted from the dataset info.

Good work retrieving the number of example from training, validation and test dataset and retrieving the number of classes from dataset_info label feature.

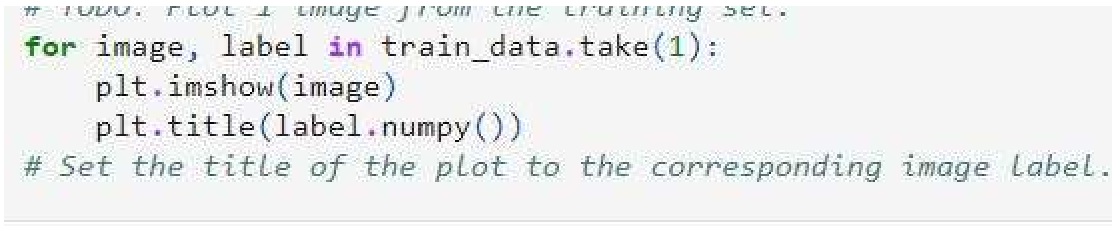
The shape of the first 3 images in the training set is printed using a `for` loop and the `take()` method.

Great job here! It's interesting to note here we have images of various sizes.

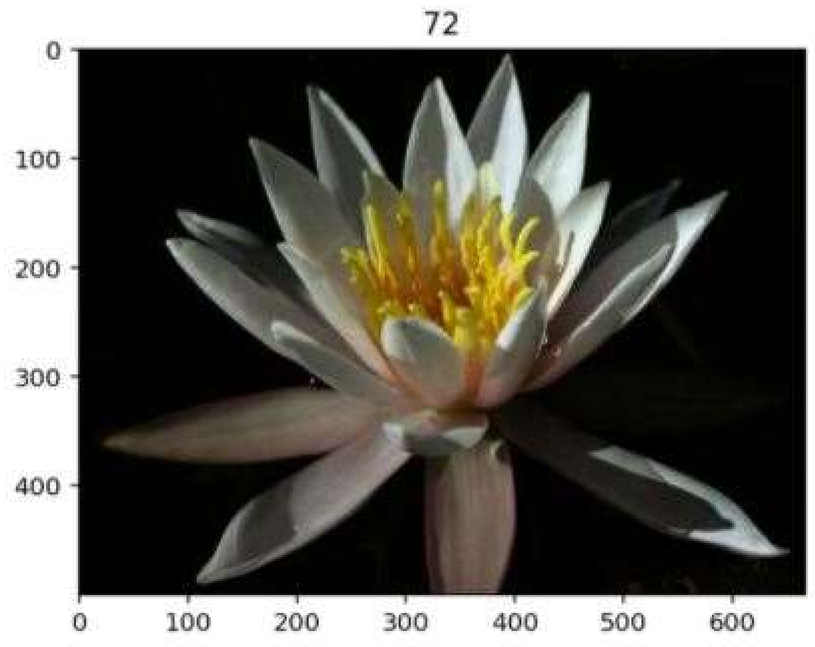
shape (500, 667, 3) label 72
shape (500, 666, 3) label 84
shape (670, 500, 3) label 70

The first image from the training set is plotted with the title of the plot corresponding to the image label.

Well done!!



```
# TODO: Plot 1 image from the training set.
for image, label in train_data.take(1):
    plt.imshow(image)
    plt.title(label.numpy())
# Set the title of the plot to the corresponding image label.
```



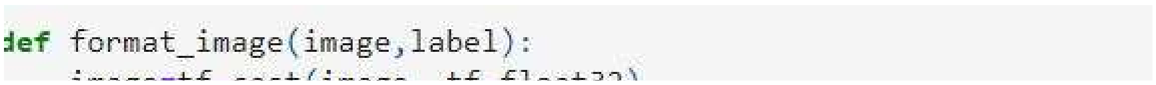
The first image from the training set is plotted with the title of the plot corresponding to the class name using label mapping from the JSON file.

Here we need to increment the label index by 1 to map class name. Please note that labels are 0 indexed and class names are 1 indexed.

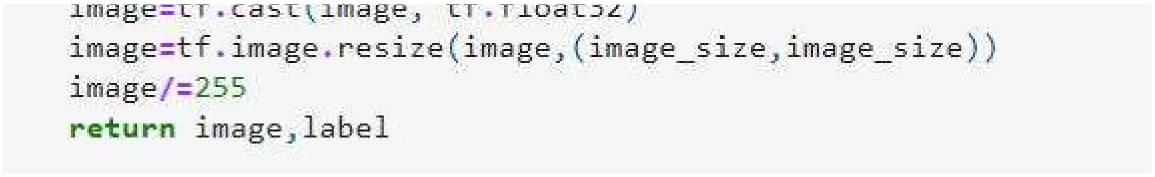
```
title=class_names[str(label.numpy()+1)]
```

The training, validation, and testing data is appropriately resized and normalized.

Well done resizing the image to 224X224 and dividing by 255 to normalize values between 0 and 1.



```
def format_image(image,label):
    image=tf.cast(image, tf.float32)
```



```
    image=tf.cast(image, tf.float32)
    image=tf.image.resize(image,(image_size,image_size))
    image/=255
    return image,label
```

A pipeline for each set is constructed with the necessary transformations.

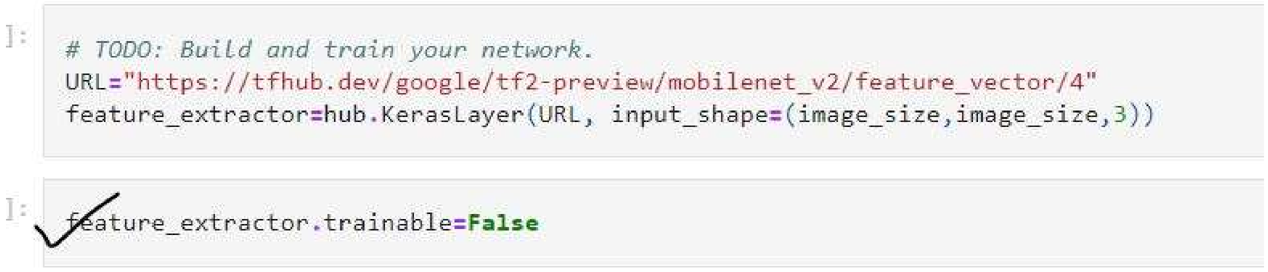
Good work building the pipeline of transforms applying shuffling, batching and normalization!!

The pipeline for each set should return batches of images.

Well done!! .batch(batch_size) is applied to all three datasets.

The pre-trained network, MobileNet, is loaded using TensorFlow Hub and its parameters are frozen.

mobilenet_v2 pretrained model is loaded from tensorflow hub and parameters of feature extractor are frozen.

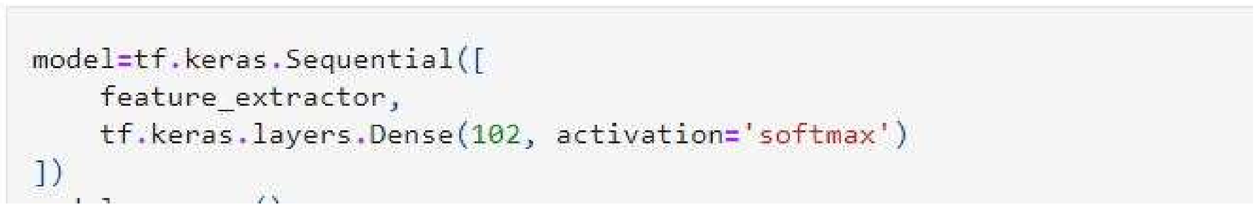


```
]:  # TODO: Build and train your network.
    URL="https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4"
    feature_extractor=hub.KerasLayer(URL, input_shape=(image_size,image_size,3))

]:  feature_extractor.trainable=False
```

A new neural network is created using transfer learning. The number of neurons in the output layer should correspond to the number of classes of the dataset.

Good work!! Number of neuron in output layer matches the number of classes.

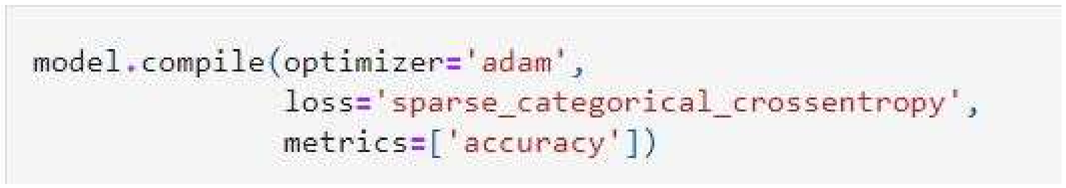


```
model=tf.keras.Sequential([
    feature_extractor,
    tf.keras.layers.Dense(102, activation='softmax')
])
```

The model is configured for training using the `compile` method with appropriate parameters. The model is trained using the `fit` method and incorporating the validation set.

trained using the `fit` method and incorporating the validation set.

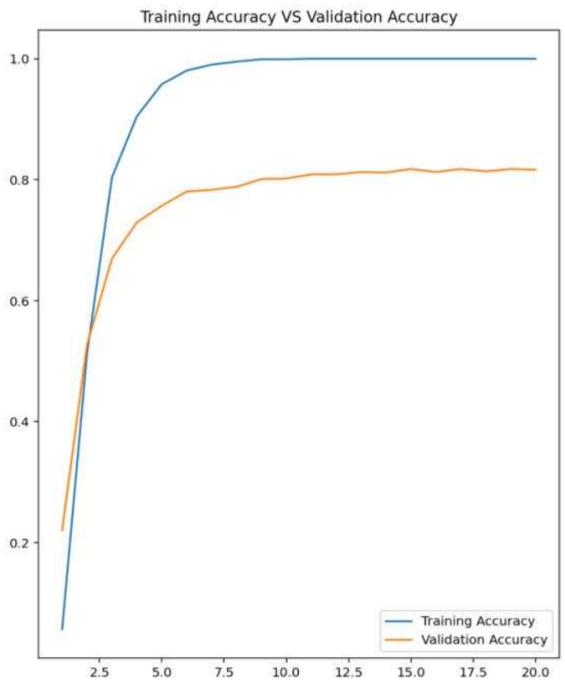
Good choice for optimizer, loss and metrics in compile method. Well done fitting model.

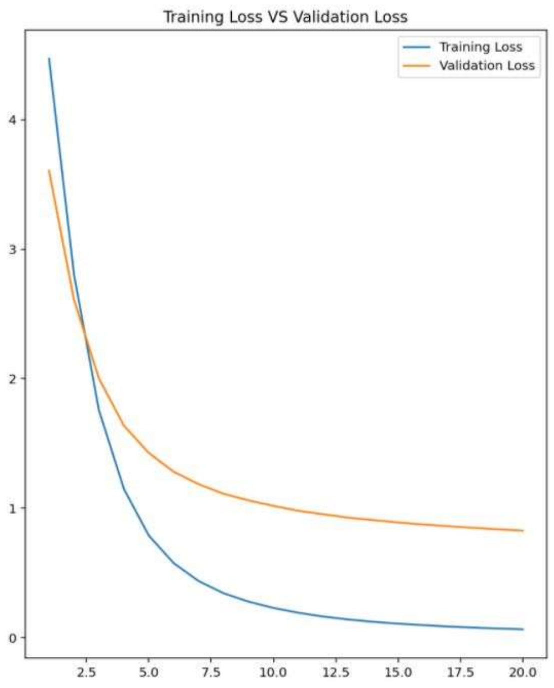


```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

The loss and accuracy values achieved during training for the training and validation set are plotted using the `history` dictionary return by the `fit` method.

Good work!! Plot shows model convergence for Loss and Accuracy curves.





The network's accuracy is measured on the test data.

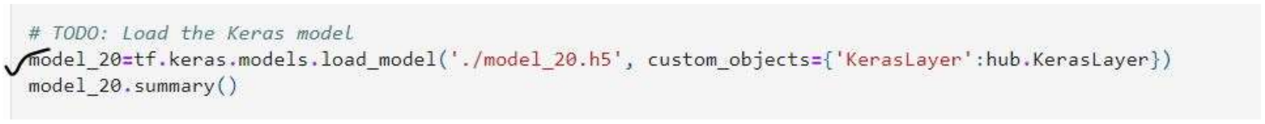
Well done achieving test accuracy of 0.7767!! This score can be further improved by training more epochs.

The trained model is saved as a Keras model (i.e. saved as an HDF5 file with extension `.h5`).

Good work saving trained model parameters as HDF5 file.

The saved Keras model is successfully loaded.

Good work!!



```
# TODO: Load the Keras model
model_20=tf.keras.models.load_model('./model_20.h5', custom_objects={'KerasLayer':hub.KerasLayer})
model_20.summary()
```

The `process_image` function successfully normalizes and resizes the input image. The image returned by the `process_image` function should be a NumPy array with shape `(224, 224, 3)`.

Good implementation ensuring below steps

- Converting to tensor
- Resizing the image to 224X224
- Normalizing each pixel dividing it by 255
- Converting image back to numpy format

The `predict` function successfully takes the path to an image and a saved model, and then returns the top K most probably classes for that image.

Good work on predict function!! Well done incrementing class index by 1.

```
class_name.append(class_names[str(num+1)])
```

A `matplotlib` figure is created displaying an image and its associated top 5 most probable classes with actual flower names.

Nice work on matplotlib visual on flower images. Prediction result depicts highest probabilities for these flower images.

- cautleya_spicata.jpg
- hard-leaved_pocket_orchid.jpg
- orange_dahlia.jpg
- wild_pansy.jpg

Part 2 - Command Line Application

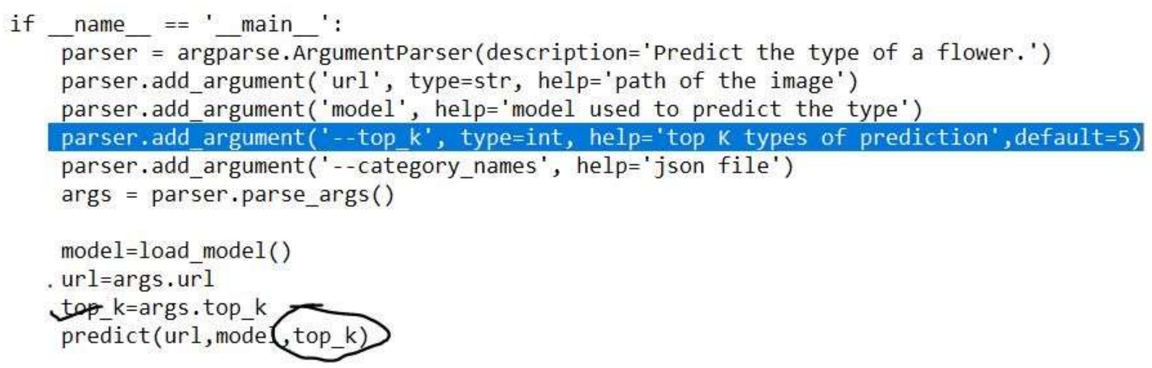
Part 2 - Command Line Application

The `predict.py` script successfully reads in an image and a saved Keras model and then prints the most likely image class and it's associated probability.

Good work on predict function!!

The `predict.py` script allows users to print out the top K classes along with associated probabilities.

Well done!! allowing user to input the topk value from command line.

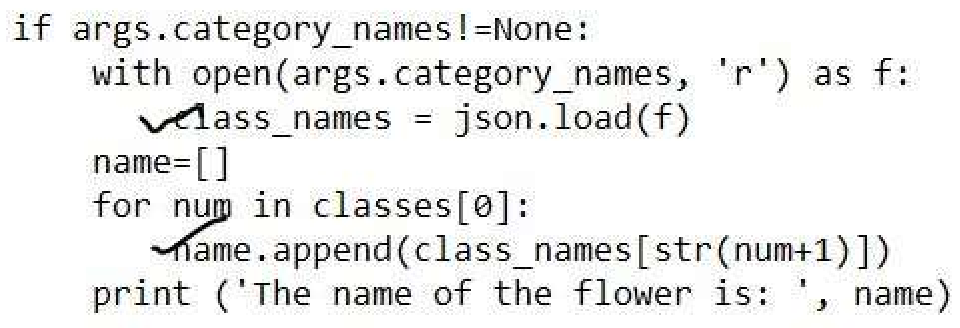


```
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Predict the type of a flower.')
    parser.add_argument('url', type=str, help='path of the image')
    parser.add_argument('model', help='model used to predict the type')
    parser.add_argument('--top_k', type=int, help='top K types of prediction',default=5)
    parser.add_argument('--category_names', help='json file')
    args = parser.parse_args()

    model=load_model()
    url=args.url
    top_k=args.top_k
    predict(url,model,top_k)
```

The `predict.py` script allows users to load a JSON file that maps the class values to other category names.

Well done loading the category_names into JSON object and mapping class index in dictionary to retrieve the flower name.



```
if args.category_names!=None:
    with open(args.category_names, 'r') as f:
        class_names = json.load(f)
    name=[]
    for num in classes[0]:
        name.append(class_names[str(num+1)])
    print ('The name of the flower is: ', name)
```

下载项目

返回 PATH

给这次审阅打分

开始